Energy-Efficient LDPC Decoder using DVFS for binary sources

S.karthikeyan[1], S.Jayashri[2]
[1]Sathyabama university/Dept of Electronics and communication Engineering, Chennai, India
Email: urkarthikeyan@rediffmail.com
[2] Adhiparasakthi Engineering college/Principal, Melmaruvathur,TN,India
Email: jayaravi2010@gmail.com

Abstract- **This paper deals with reduction of the transmission power usage in the wireless sensor networks. A system with FEC can provide an objective reliability using less power than a system without FEC. We propose to study LDPC codes to provide reliable communication while saving power in the sensor networks. As shown later, LDPC codes are more energy efficient than those that use BCH codes. Another method to reduce the transmission cost is to compress the correlated data among a number of sensor nodes before transmission. A suitable source encoder that removes the redundant information bits can save the transmission power. Such a system requires distributed source coding. We propose to apply LDPC codes for both distributed source coding and source-channel coding to obtain a two-fold energy savings. Source and channel coding with LDPC for two correlated nodes under AWGN channel is implemented in this paper. In this iterative decoding algorithm is used for decoding the data, and it's efficiency is compared with the new decoding algorithm called layered decoding algorithm which based on offset min sum algorithm. The usage of layered decoding algorithm and Adaptive LDPC decoding for AWGN channel reduces the decoding complexity and its number of iterations. So the power will be saved, and it can be implemented in hardware.**

Index Terms- **LDPC codes, Distributed Source coding, Source Channel coding, iterative decoding algorithm, layered decoding algorithm.**

I. Introduction

A. Low-Density-Parity Check-LDPC Codes:

Low-density parity-check (LDPC) codes are a class of linear block LDPC codes. The name comes from the characteristic of their parity-check matrix which contains only a few 1's in comparison to the amount of 0's. Their main advantage is that they provide a performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding. LDPC codes have performance exceeding, in some cases, that of turbo codes, with iterative decoding algorithms which are easy to implement (with the per-iteration complexity much lower than the per-iteration complexity of turbo decoders), and are also parallelizable in hardware. There are other potential advantages to LDPC codes as well. In a very natural way, the decoder declares a decoding failure when it is unable to correctly decode, whereas turbo decoders must perform extra computations for a stopping criterion (and even then, the stopping criterion depends upon a threshold that must be established, and the stopping criterion does not establish that a codeword has been found). Also, LDPC codes of almost any rate and block length can be created simply by specifying the shape of the parity check matrix, while the rate of turbo codes is governed largely by a puncturing schedule, so flexibility in rate is obtained only through considerable design effort. Also, since the validity of a codeword is validated if its parity checks, even if errors occur, they are almost always detected errors (especially for long codes). As an additional boon on the commercial side, LDPC codes are not patent protected. On the negative side, LDPC codes have a significantly higher encode complexity than turbo codes, being generically quadratic in the code dimension, although this can be reduced to some extent. Also, decoding may require many more iterations than turbo decoding, which has implications for latency.

B. Constructing LDPC Codes:

Several different algorithms exists to construct suitable LDPC codes. Only binary LDPC codes are considered here. We use N to denote the length of the code and K to denote its dimension and M = N-K. Since the parity check matrices we consider are generally not in systemic form, we usually use the symbol A to represent parity check matrices, reserving the symbol H for parity check matrices in systematic form. Following the general vector m is a K x 1 vector; a codeword is a N x 1 vector. The generator matrix *G* is N x K and the parity check matrix A is (N-K) x N , such that HG=0.

We denote the rows of a parity check matrix as

$$A = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_M^T \end{bmatrix}.$$

The equation $a_i^T c= 0$ is said to be a linear parity-check constraint on the codeword c. We use the notation $z_m = a_m^T c$ and call z_m a parity check or, more simply, a check. For a code specified by a parity check matrix A, it is expedient for encoding purposes to determine the corresponding generator matrix G. A systematic generator matrix may be found as follows. Using Gaussian elimination with column pivoting as necessary (with binary arithmetic) determine an

$$H = A_p^{-1} A = [I \quad A_2].$$

If such a matrix A , does not exist, then A is rank deficient, r = rank(A) < M . In this case, form H by truncating the linearly dependent rows from $A_p^{-1}A$. The corresponding code has R = K / N > (N - M) / N , so it is a higher rate code than the dimensions of A would suggest) Having found H, from

$$G = \begin{bmatrix} A_2 \\ I \end{bmatrix}.$$

Then HG=0, so A_pHG = AG = 0, so G is a generator matrix for A. While *A* may be sparse neither the systematic generator G nor H is necessarily sparse. Fewer than half of the elements are nonzero. Since the A matrix is sparse, it can be represented efficiently using lists of its nonzero locations. In this notation, bits are typically indexed by n or n' and the checks are typically indexed by m or m'. The set of bits that participate in check z_m (i.e., the nonzero elements on the mth row of A is denoted

$$\mathcal{N}_m = \{n : A_{mn} = 1\}.$$

Thus we can write the mth check as

$$z_m = \sum_{n \in \mathcal{N}_m} c_n.$$

The set of bits that participate in z_m except for bit ***n*** is denoted

$$\mathcal{N}_{m,n} = \mathcal{N}_m \backslash n.$$

The notation I*Nm*I indicates the number of elements in the set N_m.These sets should be considered ordered lists, with the ith element of N_m being indicated by N_m (i). The set of checks in which bit c_n participates (i.e., the nonzero elements of the nth column of *A)* is denoted

$$\mathcal{M}_n = \{m : A_{mn} = 1\}.$$

II. EXISTING METHODOLOGY

A. Distributed Source Coding Between Two Correlated Sources:

Consider a communication system of two statistically dependent signals, *X*1 and *X*2. The dependency between *X*1 and *X*2 can be fully described by the conditional probability mass function *P*[*X*1|*X*2]. A suitable source encoder that removes the redundant information bits, reduces both the length of the transmitting information and the power consumption. The correlation between signals *X*1 and *X*2 can be modelled as the input and output of a binary symmetric channel with the crossover probability of *P*[*X*1 '" *X*2|*X*1] = *p*. According to Slepian-Wolf theorem. The output of two correlated sources that do not communicate can be compressed with the same rate as if they were communicating. This is true when the decoder has access to the both compressed outputs. Such a system requires distributed source coding. The Slepian-Wolf rate region can be expressed as following:

$$R_{X_1} \geq H(X_1|X_2) \quad R_{X_2} \geq H(X_2|X_1)$$

$$R_{X_1} + R_{X_2} \geq H(X_1, X_2).$$

We study the asymmetric distributed source coding scenario in which it is assumed the signal *X*1 is compressed conventionally and sent at full rate *RX*1 e" *H*(*X*1), and is recovered perfectly at the decoder. We want to compress the *X*2 as close as possible to the Slepian-Wolf bound of *H*(*X*2|*X*1). As we can see the chosen rates for *RX*1 and *RX*2 satisfy the inequalities of above equations.

The corresponding sensor communication is shown in figure 1.

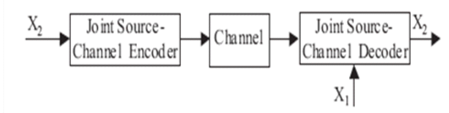

Fig.1. System for compression of X2 with the side information of X1 at the Decoder

B. Encoding and decoding with LDPC codes:

A low-density parity-check (LDPC) code is determined by its parity-check matrix (H) or, equivalently by its bipartite graph. An ensemble of LDPC codes is described by the degree distribution polynomials ë(x), and ñ(x). The bipartite graph is used in the message-passing decoding algorithm.

Encoding- Given, H to encode, i.e., compress, an arbitrary binary input sequence, we multiply X with H and find the corresponding syndrome. Z1 [length (n-k)] equivalently, in the bipartite graph, this can be viewed as binary addition of all the variable node values that are connected to the same check node.

Decoding algorithm- The decoder must estimate the n-length sequence *X* from its (n — k) long syndrome *Z1* and the corresponding n-length sequence Y

We use the following notation:

- x_i, y_i £ {0, 1}, *i* = 1, 2, ..., n, are the current values of Xi and *Y*i. respectively, corresponding to the i th variable node vi:
- Li£ {2, 3, ...},i< = 1,2, ..., *n,* is the degree of V_i
- ,$q^{out}_{i,m}$ ($q^{in}_{i,m}$)£ R, i= 1, 2, ..., n, m = 1, 2,....l_i is the log-likelihood ratio (LLR) sent along the mth edge *from (to)* V_i,
- *Sj* £ {0, 1} *j* = 1, 2, ..., *n* — *k,* is the value of *Z1,j* corresponding to the Jth check node C_j, i.e., the j th syndrome component;
- r_j £ {2, 3, ...}, *j* = 1, 2, ..., *n* — *k,* is the degree of c_j;
- t^{out}_{jm} ($t^{in}_{j,m}$)£ e R, j = 1, 2, ..., n-k, m = 1, 2, ...rj, is the LLR sent along the mth edge *from (to)* c_j.

Step 1: Setting

$$q_{i,0} = \log\frac{\Pr[x_i = 0|y_i]}{\Pr[x_i = 1|y_i]} = (1-2y_i)\log\frac{1-p}{p}$$

Step 2: i=1,2…..n, p<=0.5 the LLR sent from the i th variable node vi along the m th edge is

$$q_{i,m}^{\text{out}} = q_{i,0} + \sum_{j=1,\, j\neq m}^{l_i} q_{i,j}^{\text{in}}$$

M=1,2,.....li, i=1,2,....n where initially $q^{\text{in}}_{i,j=0}$

step 3: The values qouti,m are assigned to the corresponding tinj,ð (i,m,j) according to the connections in the bipartite graph and are then used to do the processing at the check nodes. From the "tanh rule" and the syndrome information, the LLR sent from the jth check node cj along the mth edge is

$$\tanh\left(\frac{t_{j,m}^{\text{out}}}{2}\right) = (1-2s_j)\prod_{i=1,\, i\neq m}^{r_j}\tanh\left(\frac{t_{i,m}^{\text{in}}}{2}\right)$$

j= 1, 2, ..., n-k, m = 1, 2, ...r_j, The inclusion of the (1 — 2sj) factor accounts for the syndrome information.

step 4: Now $q^{\text{in}}_{i,m} = t^{\text{out}}_{j,ð\,(i,m,j)}$ for all edges in the bipartite graph, which can be used to start a new iteration and estimate xi from

$$\hat{x}_i = \begin{cases} 0, & \text{if } q_{i,0} + \sum_{m=1}^{l_i} q_{i,m}^{\text{in}} \geq 0 \\ 1, & \text{if } q_{i,0} + \sum_{m=1}^{l_i} q_{i,m}^{\text{in}} < 0. \end{cases}$$

III. PROPOSED METHODOLOGY

A. LDPC Layered Decoding Based on Offset MIN-SUM Algorithm:

In practice, the LDPC decoder is typically set to run for data convergence until a prescribed maximum number of iterations (e.g. 20) depending on the code rate. However, the actual number of decoding iterations varies from frame to frame. In the case that channel data comes in constant time interval, a conventional decoder has to be configured to accommodate the worst case scenario. As a result, the decoder often remains idle since for most frames, the decoding process ends far earlier than the maximum number of iterations. Thus it is not power efficient. In the decoders proposed is based on an on-the-fly computation paradigm, optimized dataflow graphs are introduced to reduce the logic and internal memory requirements of the LDPC decoder and at the same time the decoder's parallelization is tailored to average number of decoding iterations for the target frame error rate for the operating SNR region. These decoders buffer statistically for different parallelization based on average number of decoding iterations while ensuring the performance similar to that of a fixed iteration decoder configured for maximum number of iterations. These decoders are almost fully utilized and run at the maximum frequency. The proposed paper improves the system. There have been researches on early termination of frame that cannot be decoded even if the maximum iterations are applied. In both of papers, early termination of the iterative process is determined by checking the messages during the decoding. Between initial check error and number of decoding iterations is insufficient. This paper describes a more comprehensive design that accommodates both fading channels and the more difficult case of AWGN channels. This paper presents the LDPC decoding algorithm based on offset min-sum and layered decoding.

Assume binary phase shift keying (BPSK) modulation (1 is mapped to "1 and 0 is mapped to 1) over an additive white Gaussian noise (AWGN) channel. The received values *yn* are Gaussian with mean *xn* = 1 and variance ä^(2). The Iterative two-phase message-passing (TPMP) algorithm, also known as belief-propagation (BP) algorithm is computed in two phases. One is a check node processing and the other is variable node processing. In the check node processing, each row of the parity matrix is checked to verify that parity check constraints are satisfied. In the variable node processing the probability will be updated by summing up the other probabilities from the rest of the rows and the a prior probabilities from the channel output. The message passing algorithm can be simplified to the belief-propagation based algorithm (also called Min-Sum algorithm). While greatly reducing the decoding complexity in implementation, the Min-Sum degrades the coding performance. The improved BP based algorithm, Normalized-Min-Sum and Offset-Min-Sum eliminates this performance degradation. Following the same notation in [8], the check node processing can be expressed as:

$$R^{(i)}_{mn} = ä^{(i)}_{mn} \max\,(ê^{(i)}_{mn} \text{ “ } â,\, 0)$$

$$ê^{(i)}_{mn} = \%R^{(i)}_{mn}\% = \min\ \%\, Q^{(i”1)}_{n’m}\%$$

$$n”\text{N}(m)\backslash n$$

Where $Q^{(i)}_{nm}$ is the message from variable node *n* to check node *m*,

$R^{(i)}_{mn}$ is the message from check node *m* to variable node *n*, and superscript *i* denotes the *i*th iteration. *M*(*n*) is the set of the neighbouring check nodes for variable node *n*, and *N*(*m*) is the set of the neighbouring variable nodes for check node *m*, *â* is a positive constant and depends on the code parameters

The sign of check-node message $R^{(i)}_{mn}$ is defined as

$$ä^{(i)}_{mn} = (\quad \text{Ï} \quad \text{sgn}\,(Q^{(i”1)}_{n’m}))$$
$$n\_”\text{N}(m)\backslash n$$

In the variable node processing,

$$Qn = L^{(0)}_{n} + \text{ “ } \text{R}^{(i)}_{mn}$$

$$m”M(n)/m$$

Where the log-likelihood ratio of bit n is $L^{(0)}_{n} = y_{n.}$ For final Decoding

$$P_n = L^{(0)}_{n} + \text{ “ } \text{R}^{(i)}_{mn}$$
$$m”M(n)$$

A hard decision is taken by setting $x_n = 0$ if $P_n(x_n)$ *e"* 0, and $x_n = 1$ if $P_n(x_n)$ *d"* 0. If xHT = 0, the decoding process is finished with ˆ*xn* as the decoder output; otherwise, repeat processing of Equation. If the decoding process does not end within predefined maximum number of iterations, itmax,

DOI: 01.IJIT.01.01.37

stop and output an error message flag and proceed to the decoding of the next data frame. Mansour, introduces the concept of turbo decoding message passing (also called as layered decoding) for their AA-LDPC codes using BCJR which is able to reduce the number of iterations without performance degradation when compared to the standard message passing algorithm. Contrast to two phase message passing algorithm, where all check-nodes are updated simultaneously in each iteration, layered decoding view the *H* matrix as a concatenation of *j* = *dv* sub-codes, The *H* matrix is divided into different block-rows and block columns. After the check-node processing of one layer, the updated messages are immediately used to calculate the variable node message, whose results are then applied to next layer of sub-code. Each iteration in the layered decoding algorithm is composed of *j* sub-iterations. The processing of one block row is called a sub-iteration and each iteration is composed of *j* = *dv* sub-iterations.

Mathematically, the layered decoding algorithm can be described as in

```
Algorithm:
    step1: $R^{(0)}_{l,n} = 0$, “   l” [1, dv],
                               n “ [1, dc]
    step2: $P_n = L^{(0)}_n$ , “ n “ [1, dc]
    step3: for each  i = 1, 2, · · · , itmax
                                           do
    step4: for each  l = 1, 2, · · · , dv do
    step5: for each  n = 1, 2, · · · , dc do
        $Q^{(i)}_{l,n=} P_n R^{(i”1)}_{l,n}$
        $R^{(i)}_{l,n} = f\,[Q(i\,)]$  “n_ “ [1, dc]
        $P_n = Q^{(i)}_{l,n} + \_R^{(i)}_{l,n}$.
```

IV. RESULT

A. Source Coding for LDPC with Two correlated nodes by using SUM Product and Layered Decoding Algorithm

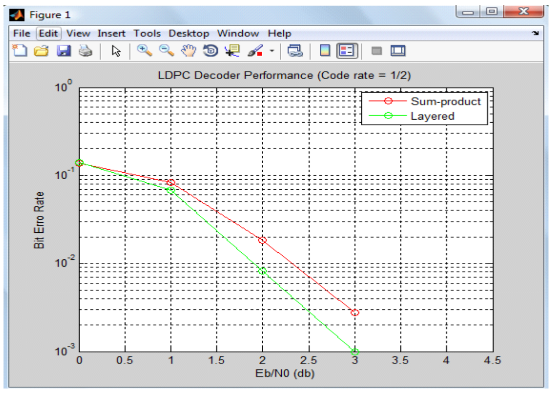

Fig.4. The performance of LDPC decoder by using sum product and layered decoding algorithms

The results showed how LDPC codes can be used in providing reliable data transmission and developing aggregation techniques for correlated data in wireless sensor networks.. It was shown that while some FEC coding schemes can improve the energy efficiency of a communication link, it proves that FEC using LDPC codes can reduce the transmission power. The simulation results showed that using LDPC codes for FEC is significantly more efficient than using BCH codes.

V. CONCLUSION

The ‘Source and channel coding’ and ‘distributed Source coding‘ of two correlated nodes are done. The performances of Both are compared. And Source and channel coding with LDPC for two correlated nodes under AWGN channel is implemented in this paper. In this iterative decoding algorithm is used for decoding the data, and it’s efficiency is compared with the new decoding algorithm called layered decoding algorithm which based on offset min sum algorithm. From this we can conclude that by using layered decoding algorithm the decoding complexity is much reduced and its performance is improved when compared with iterative decoding algorithm.

REFERENCE

[1] A. D. Liveris, Z. Xiong, and C. N. Georghiades, “Compression of binary sources with side information at the decoder using LDPC codes,” *IEEE Comm. Letters*, vol. 6, pp. 440–442, Oct. 2002.

[2] T. J. Richardson and R. L. Urbanke, “The capacity of low-density paritycheck codes under message passing alorithm,” *IEEE Trans. Inform,Theory*, vol. 47, pp. 599–618, Feb. 2001.

[3] S. S. Pradhan and K. Ramchandran, “Distributed source coding usingsyndromes (DISCUS): design and construction,” *Proc. IEEE Data Compression Conference*, pp. 158–167, March 1999.

[4] S. S. Pradhan and K. Ramchandran, “Distributed source coding: symmetric rates and applications to sensor networks,” *Proc. IEEE Data Compression Conference*, pp. 363–372, March 2001.

[5] Weihuang Wang, Gwan Choi and Kiran K. Gunnam “Low-Power VLSI Design of LDPC Decoder Using DVFS for AWGN Channels” 22nd International Conference on VLSI Design. 2009

[6] http://books.google.com/books?id=9JVsRYL0ZkUC&pg=PA206&dq=bit+flipping+algorithm++%2Bldpc&ei=net7S92_N5D8lAN5D8lATzl8izCA&cd=6#v=onepage&q=bit%20flipping%20algorithm%20%2Bldpc&f=false

[7] http://books.google.com/books?id=9EJQk0mPlYC&pg=PA651&dq=offset+minsum+decoding+algorithm%2Bldpc&ei=GvWXS__dM5DMlQSU06dM5DMlQSU06zxCQ&cd=1#v=one page&q=&f=false

[8]http://books.google.com/books?id=z8nmMkUFqdwC&pg=PA534&dq=offset+minsum+decoding+algorithm%2Bldpc&ei=GvWXS__dM5DMlQSU06zxCQ&cd=2#v=onepage&q=&f=false